SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

LADONTAROBOTIN ETÄOHJAUS

TEKIJÄ/T: Pekka Thil

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Pekka Thil	
Työn nimi Mosaiikin Ladontarobotin Etäohjaus	
Päiväys 30.4.2013	Sivumäärä/Liitteet 25
Ohjaaja(t) Risto Niemi	
Toimeksiantaja/Yhteistyökumppani(t)	

Tiivistelmä

Opinnäytetyö tehtiin Savonian Ammattikorkeakoulun Varkauden kampuksella. Työn aiheena oli luoda etäohjausohjelmisto B-rakennuksen alakerrassa sijaitsevalle mosaiikinladontarobotille. Työ tehtiin käyttäen Java-ohjelmointikieltä.

Ohjelmisto koostuu kolmesta osasta: Käyttöliittymä, johon kuuluu html-sivu ja Java Servlet –luokka, Server-client Socket tyyppinen palvelin, sekä kontrolleri, joka välittää tiedon robotille. Html-sivu sisältää useita painikkeita, jotka suorittavat servletillä olevia toimintoja. Palvelin yhdistää asiakkaan robottiin kytkettyyn tietokoneeseen. Kontrolleri muuntaa palvelimelta tulevat käskyt robotin ymmärtämään muotoon ja välittää ne robotille käyttäen sovellettua clientsockettia.

Opinnäytetyö käsittelee ladontarobotin ominaisuuksia ja laitteistoa, käytettyjä ohjelmistoja sekä Java-ohjelmointikieltä ja historiaa. Mukana on myös hiukan robotiikan historiaa. Java-kieltä ja sen rakennetta tarkastellaan ohjemoijan näkökulmasta.

Työn lopputuloksena on ohjelmisto, jolla ladontarobottia voidaan ohjata internetin välityksellä.

Avainsanat
Robotti, Etäohjaus, Java, Ohjelmistosuunnittelu

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Pekka Thil			
Title of Thesis Remote Control for Mosaic Assembly Robot			
Date	30.4.2013	Pages/Appendices	25
Supervisor(s) Risto Niemi			
Client Organisation /Partners			

Abstract

The thesis was made at the campus of Savonia University of Applied Sciences in Varkaus. The purpose of the thesis was to create remote control software for mosaic assembly robot located in the bottom floor of the B-wing.

The software consists of three parts: interface that includes html-page and Java Servlet -class, server-Client socket –type server and Controller that transfers data to the robot. Html-page contains multiple buttons that execute methods from servlet. Server connects client to the computer that is connected to the robot. Controller translates messages coming from the server in to the format that the robot understands and delivers messages to the robot via customized client socket.

The thesis covers the the specifications and hardware of the assembly robot, used software and Java-programming language and its history. Little bit of history of robotics is also included. Java-languange is described from programmer's point of view.

The final result of the thesis is software that can be used to control assembly robot via internet.

Keywords
Robot, Remote Control, Java, Software Development

SISÄLTÖ

LÄHTEET JA TUOTETUT AINEISTOT

LIITTEET

1 JOHDANTO

Robottien käyttö niin Suomessa kuin maailmalla on yleistynyt kymmenen viime vuoden aikana huomattavasti. Monet teollisuusalan yritykset ovat siirtyneet perinteisestä ihmistyövoimasta automatisoituihin linjastoihin tehokkuuden kasvattamiseksi.

Robottien etäohjaus on vielä toistaiseksi ollut vähäistä. Etäohjauksella on omat hyötynsä, mutta myös riskinsä ja puuttensa, varsinkin kun ohjattavana on usean kilon painoinen, jopa useita metrejä sekunnissa liikkuva robottikäsi. Roboteilla harvemmin on ns. omaa älyä, joten ne tarvitsevat ihmisen pitämään huolta ettei vahinkoja synny. Etäohjauksen avulla ihmisen ei kuitenkaan tarvitsisi olla koko aikaa robotin lähettyvillä vaan voisi hallita robottia lähes mistä päin maailmaa hyvänsä.

Ohjelmistolla voidaan valita ja ajaa ohjelmia, joita robotille on tallennettu. Koska kyseessä on prototyyppi tason ohjelmisto, on joitain ominaisuuksia jätetty pois, kuten robotin toimintanopeuden säätäminen ja nivelten venyttely.

Opinnäytetyössä käsitellään pääasiallisesti etäohjausohjelmiston rakennetta ja toteutusta RV-2AJ -robottikädelle. Opinnäytetyössä käydään läpi työssä käytettyä laitteistoa ja ohjelmistoa sekä Java-ohjelmointikielen rakenteita ja historiaa.

.

2 ETÄOHJAUKSESTA

Etäohjauksen historia ulottuu 1800-luvun loppuun vuoteen 1898, jolloin henkilö nimeltään Nikola Tesla demonstroi Madison Square Gardenissa pidetyssä näyttelyssä ensimmäistä kertaa laitteen, jota pystyi ohjaamaan ilman fyysistä kontaktia. Kyseinen laite oli veneen pienoismalli, jota pystyi ohjaamaan radioaaltojen taajuutta muuttamalla. Sen jälkeen ihmiskunta on keksinyt useita tapoja kauko-ohjata laitteita. Yksi tunnetuimpia etäohjausta soveltavia laitteita on kaukosäädin. Useimmat niistä toimivat infrapunaohjauksella, jolloin ohjaimessa oleva pieni infrapunavalaisin lähettää valonsäteen, joka sisältää binäärimuotoisen pulssijonon (käskyn), jonka vastaanotin lukee ja suorittaa.

Pitkän matkan etäohjaus on vielä toistaiseksi melko harvinaista, mutta on vähitellen yleistymässä. Internetin käyttö etäohjauksessa on alkanut yleistymään huomattavasti. Esimerkiksi Rakennus- ja kotiautomaatioon erikoistunut Ouman Oy mainostaa etäohjausohjelmistoa EH-net, jolla voidaan säätää mm. ilmanvaihtoa ja lämmitystä internetin välityksellä.(Ouman)

3 ROBOTIIKAN HISTORIAA

Ensimmäisiä robottien kaltaisia koneita alkoi esiintyä jo kauan ennen teollista aikakautta. Yhden ensimmäisistä toimivista roboteista, joka oli huilua soittava robotti, rakensi Jacques de Vaucanson vuonna 1738. Toinen Jacquesin rakentama robotti, joka on ehkä tunnetumpi, oli samana vuonna esitelty mekaaninen ankka, joka vaakkui, räpytteli siipiään ja omasi jopa mekaanisen ruuansulatuselimistön. (A brief history of Robotics)

Varsinaiset teolliset robotit alkoivat kuitenkin yleistyä vasta 1960-luvun alussa, kun ensimmäiset teollisuusrobotit saatiin toimintaan. Ensimmäinen käsivarsirobotti, nimeltään Unimate, valmistettiin vuonna 1962 General Motorsille tekemään vaarallisia ja yksitoikkoisia töitä. Varsinkin 1980-luvun alkupuolella teollisuusrobotit yleistyivat erittäin nopeasti. Yksi syy robottien yleistymiselle on niiden kustannukset verrattuna ihmisvoiman käytön kustannuksiin. Vielä vuonna 1990 robottien ja työvoiman kustannukset olivat suurinpiirtein samat, mutta jo 1992 robotit olivat n. puolet halvempia käyttää kuin ihmiset. (CRAIG, 1-2)

4 LAITTEISTO

4.1 Mitsubishi RV-2AJ

Rv-2AJ on teollisuuskäyttöön tarkoitettu kompakti robottikäsi. Se on suunniteltu tarkkuutta ja joustavuutta vaativiin töihin. Sen kantavuus on maksimissaan 2 kg ja sen ulottuvuus on maksimissaan n. 480 mm ilman työkalua ja kaikki nivelet suoristettuna. Todellinen ulottuvuus on kuitenkin vain 410 mm. Robotin alustan kääntösäde on 300 astetta ja robotin alimman nivelen (J2) kääntyvyys on 180 astetta, keskimmäisen nivelen (J3) 230 astetta ja uloimmaisen nivelen (J5) 180 astetta. Lisäksi työkalua voi pyörittää akselin (J6) ympäri 400 astetta. Liikkeiden tarkkuus vakaalla pohjalla on ±0,02 mm. (Mitsubishi, 2-5)

Kuva 1: Mitsubishi RV-2AJ -ladontarobotti

4.1.1 Melfa BASIC

Melfa BASIC on ohjelmointikieli, jolla robotille annetaan käskyjä. Se pohjautuu nimensä mukaisesti Basic-ohjelmointikieleen. Ohjelma rakentuu riveistä, joille ohjelmoija antaa rivinumerot. Jokaisella rivillä voi olla vain yksi komento. Ohjelman koodirivit suoritetaan numerojärjestyksessä ylhäältä alas. Yleisimmin käytetään kymmenellä kerrottuja lukuja (10,20,30,40…), jolloin rivien väliin voidaan lisätä koodirivejä tarvittaessa jopa yhdeksän kappaletta.

Melfa BASIC käyttää yksin kertaisia komentoja esim. 10 MOV P1 liikuttaa robotin tallennettuun pisteeseen P1. Robotin nopeutta voidaan hallita komennolla SPD 10, joka tässä esimerkissä muuttaa nopeuden 10 prosenttiin maksiminopeudesta. Huomattavaa on,

että ohjausyksikössä asetettu nopeus on koodissa aina maksiminopeus eli spd 100. Esim. jos ohjausyksikön asettama nopeus on 50 ja koodissa nopeus pudotetaan 10 prosenttiin, todellinen nopeus on 5 prosenttia.

Paikkakoordinaatit asetetaan opetusyksiköllä tai tietokoneen kautta. Koordinaatit tallennetaan samalla nimellä kuin ohjelma, jotta robotti tietää mihin asentoihin käskyillä viitataan.

Työkaluja hallitaan käyttämällä magneettiventtiileitä, joille voidaan antaa vain kaksi arvoa: on (1) tai off (0). Magneettiventtiileitä käytetään ohjelmassa M_OUT()-komennolla. Parametrinä annetaan venttiilin numero ja arvoksi annetaan 1 tai 0.

Ohjelma päätetään aina komennolla END.

4.1.2 Ohjausyksiköt

4.1.2.1 CR1-ohjausyksikkö

CR1 on robotin ohjausyksiköiden keskus. Sillä ohjataan kaikkia tärkeimpiä toimintoja mm. ajettavan ohjelman valinta, käynnistys, suoritusnopeus ja pysäytys. Ohjausyksikkö sisältää Ethernet-väylän, jonka kautta ohjausyksikköä voidaan hallita esim. tietokoneella. Yksikkö sisältää myös törmäysanturin, joka pysäyttää ohjelman törmäyksen sattuessa. CR1:en on myös kytketty opetusyksikkö, joka aktivoidaan kääntämällä ohjausyksikön avainkytkin asentoon ”teach”.

Kuva 2: Robotin Ohjausyksiköt. CR1-ohjausyksikkö (alimmainen), E1070-operointiyksikkö (keskimmäinen) ja E710-operointiyksikkö (ylin)

4.1.2.2 Opetusyksikkö

Opetusyksikkö otetaan käyttöön kääntämällä avainkytkin enable-asentoon. Opetusyksikkössä on ns. deadman-kytkin, joka estää robotin liikuttelun, jos kytkin ei ole alaspainettuna. Opetusyksikön avulla robotille voidaan mm. opettaa tärkeitä koordinaatteja, kääntelemään robotin akseleita Jog-toiminnolla tai kirjoittaa lyhyitä ohjelmia.

4.1.2.3 E1070-operointiyksikkö

E1070 on kytketty CR1-ohjausyksikköön, jolloin voidaan operointiyksikön E-Designer-ohjelmiston avulla mm. ohjata robottia ja laitesolun valoja.

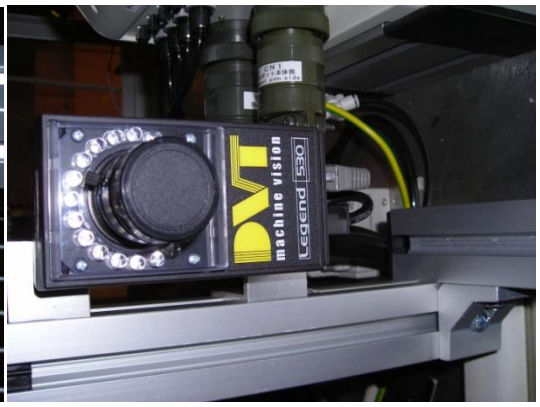
4.1.2.4 E410-operointiyksikkö

E410 on mustavalkea kosketusnäytöllinen operointipaneeli, jolla voidaan ohjata kuljetinta. Operointiyksikön ohjelmointi suoritetaan E-Designer-ohjelmistolla. Kuljettimella on kaksi toimintoa: Normal run, joka kuljettaa kuljettimen päällä olevat esineet valokennon kohdalle, ja Belt clear, joka tyhjentää kuljettimen.

4.1.3 Kamerat

Laitesolussa on kaksi kameraa, värikamera 542C ja harmaasävykamera 530. Kamerat on yhdistetty Ethernet-liitännän kautta tietokoneeseen. DVT 542C sijaitsee kuljettimen yläpuolella valokennon kohdalla ja sen resoluutio on 640*480 pikseliä. DVT 530 sijaitsee robotin vieressä olevan tason yläpuolella ja sen resoluutio on sama kuin värikameralla, 640*480 pikseliä.

Kuva 3: DVT 542C –värikamera



Kuva 4: DVT 530 -harmaasävykamera

5 JAVA

5.1 Historiaa

Java, joka alun perin tunnettiin nimellä Oak, kehitettiin 1990-luvun alussa. Sun Microsystems loi työryhmän, jonka tavoitteena oli luoda ohjelmointikieli kuluttajaelektroniikan tarpeisiin James Goslingin johdolla. Myyntivalttina olisi alustariippumattomuus. Työ tuotti tulosta, kun ensimmäinen kaupallinen versio Java 1.0 julkaistiin vuonna 1995. Alkuperäinen nimi Oak jouduttiin muuttamaan Javaksi kaupallisten syiden vuoksi. (Kinnunen 2008)

Javan kehitykseen on lainattu ominaisuuksia mm. C-kielistä. Kehitysideana oli, että Java-ohjelmia voidaan kehittää ja ajaa millä tahansa alustalla. Java toimii useissa eri käyttöjärjestelmissä, mm. Linuxissa ja Windowsissa.

5.2 Yleistä

Java muistuttaa melko paljon C-kieltä. Siitä löytyy perusrakenteet kuten ehto- ja toistolauseet. Lauseet päätetään puolipisteeseen ja lauseet kootaan lohkoiksi aaltosulkeita käyttämällä. Javassa käytetään luokkia ja moduuleita ja luokkien periyttämistä. (Kinnunen 2008)

Turvallisuuden takaamiseksi java-ohjelmat suoritetaan virtuaalikoneessa, joka hidastaa suoritusta hiukan, mutta on turvallisempi kuin konekieliohjelmat. Virtuaalikone kääntää java-koodia prosessorin ymmärtämiksi käskyiksi reaaliaikaisesti. Samalla virtuaalikone estää java-ohjelmaa vaikuttamasta muiden prosessien toimintaan. (Kinnunen 2008)

5.3 Muuttujat

Javassa on monia eri muuttujatyyppejä, joista yleisimmin käytettyjä ovat int, double, boolean ja string. Kokonaislukuihin käytetään int-muuttujaa, desimaalilukuihin double-muuttujaa ja merkkijonoja varten on string-muuttuja. Boolean-muuttujaa käytetään, kun tarkastellaan totuuksia. Sille voi antaa arvoiksi vain joko true (tosi) tai false (epätosi).

Ennen muuttujien käyttöä ne tarvitsee esitellä. Samoin kuin esim. C-kielessä, javassa muuttujat esitellään antamalla ensin muuttujatyyppi, seuraavaksi muuttujan nimi ja lopuksi muuttujalle voi antaa myös lähtöarvon. Esimerkiksi jos luodaan muuttuja, joka sisältää

vuosiluvun, jonka oletusarvo on 2007, muuttuja esitellään seuraavalla tavalla: int vuosi=2007;. Oletusarvon asettamisella saatetaan vähentää virheitä.

5.4 Ehtolauseet

Ehtolausen avulla voidaan suorittaa eri koodi suorittamisen eri vaiheissa. Ehtolause aloitetaan if-sanalla, jonka jälkeen annetaan itse ehto suluissa. Ehtoja voi olla yksi tai useampia. Tämän jälkeen tulee aaltosuluissa koodi, joka suoritetaan ehdon täyttyessä. Ehtolause voi myös sisältää else-osion, jonka koodi suoritetaan, mikäli ehto ei täyttynyt. Else-osio ei ole pakollinen. Tätä lauserakennetta kutsutaan else-if lauseeksi.

Toinen ehtolause on nimeltään switch-case rakenne, jota toimii paremmin monivalinnassa . Se aloitetaan switch-sanalla, jonka jälkeen annetaan ehto suluissa, joka voi olla vaikka muuttuja tai laskutoimitus. Sen jälkeen aaltosulkujen sisällä ovat case-kohdat, jotka sisältävät suoritettavat koodit. Jokaisen case-kohdan perässä on mahdollinen vastaus ehtoon. Jos ehdon arvo on sama kuin vastaus, case-kohdan koodi suoritetaan. Rakenne saattaa myös sisältää default-kohdan, joka suoritetaan jos yksikään case-arvo ei ollut sama kuin ehdon arvo.

5.5 Toistolauseet

Toistolauseita on kolmea eri tyyppiä. Nämä ovat While, Do-While ja For-toistolauseet. Toistolauseiden avulla voidaan suorittaa yhtä tai useampaa koodiriviä haluttu määrä, jolloin vältytään samojen koodirivien kirjoittamisesta useampaan kertaan. Toistolausetta käytetään esimerkiksi luettaessa tiedostosta tekstiä rivi kerrallaan. Do ja Do-while -lauseiden käyttö vaatii tiettyä huolellisuutta, jottei ohjelmoida niin sanottua ikuista silmukkaa. Ikuinen silmukka ei ole varsinainen ohjelmointivirhe, mutta ohjelman suoritus pysähtyy muilta osin ohjelman jäädessä suorittamaan silmukan sisältöä loputtomasti.

For-lausetta käytetään yleensä, kun lausetta toistetaan vakiomäärä. Lauseen rakenne on seuraavanlainen: Lause alkaa for-sanalla, jonka perään sulkeiden sisään lisätään kolme kohtaa, jotka erotetaan puolipisteellä. Ensimmäisenä asetetaan alkuarvo lukumuuttujalle, jota käytetään toistojen lukumäärän laskemiseen. Seuraavaksi asetetaan ehto toistolauseelle. Kun aiemmassa kohdassa asetettu luku ei vastaa ehdon vaatimuksiin, toisto pysäytetään. Lopuksi asetetaan etenemisarvo, jonka mukaan arvo kasvaa tai pienenee. Tällä varmistetaan toiston jatkuvuus.

While ja do-while –lauseita käytetään, kun halutaan dynaamisempi toistolause. Lauseet sisältävät ehdon, joka määrittää missä tilanteissa lauseet toistetaan. Kun ehto täytetään, toistaminen lopetetaan. Erona näillä lauseilla on niiden sisältämän ehdon tarkastussijainti. While –lauseen ehto tarkastetaan ennen koodin toistamista, jolloin on mahdollista jättää koodi toistamatta kokonaan, jos ehto täyttyy jo ennen toistokoodia. Do While –lausetta käytettäessä koodi toistetaan vähintään kerran, koska ehto tarkastetaan vasta toiston lopussa.

5.6 Operaattorit

Muuttujille voidaan tehdä myös erilaisia aritmeettisia operaatioita. Käytettävissä ovat yleisimmät matemaattiset operaatiot kuten yhteen-, vähennys-, jako-, ja kertolasku. Käytettävissä on myös jakojäännös, joka suoritetaan merkillä %. Muuttujan arvon kasvattamiseen yhdellä on olemassa kaksi tapaa. Ensimmäinen tapa on lisätä muuttujan arvoon 1 ja sijoittaa se muuttujaan itseensä esimerkiksi a=a+1. Toinen tapa arvon kasvattamiseen yhdellä on a++. Sama pätee myös vähentämiseen, jolloin käytetään + merkkien sijasta - merkkejä.

Vertailu- ja suhteellisuus-operaattoreilla voidaan tutkia muuttujien arvojen välisiä suhteita. Näitä käytetään varsinkin ehtolauseiden ehdoissa.

Taulukko 1: Vertailuoperaattorit

<	pienempi kuin
<=	pienempi tai yhtä pieni kuin
>	suurempi kuin
>=	suurempi tai yhtä suuri kuin
==	yhtä suuri
!=	erisuuri

5.7 Java Servlet

Servlettiä käytetään yleensä dynaamisten html-sivujen luomiseen kuten esimerkiksi kyselyihin. Java servlet on periaatteessa html-sivu java-koodin sisällä. Sen vastakohtana on JSP eli Java Server Pages. Jsp on html-muotoinen sivu, joka sisältää java-koodia. Näitä kahta käytetään hyvin usein yhdessä.

Servletit tarvitsevat vähintään java.io.*, javax.servlet.* ja javax.servlet.http.* kirjastot. Servletit periytyvät luokasta HttpServlet. Servletit sisältävät kaksi metodia: doPost ja doGet. Näistä molemmat yleensä suoritetaan, mutta vain toinen sisältää koodin. Molemmilla metodeilla on parametrit HttpServletRequest ja HttpServletResponse. Request käsittelee asiakkaan pyynnöt ja response palvelimen vastaukset. (Kinnunen, 6-8)

6 OHJELMISTOT

6.1 Eclipse

Eclipse on avoimen lähdekoodin projekti, jonka tarkoituksena on kehittää laajennettava kehitysympäristö. Eclipse-säätiö on voittoa tavoittelematon. Säätiö itse ei tee kehitystyötä vaan työ tehdään vapaaehtoisen työvoiman avulla avoimen koodin periaatteella. Työvoima koostuu yrityksistä ja yksityisistä ihmisistä. (Eclipse)

IBM loi Eclipse-säätiön vuonna 2001. Vuonna 2004 Eclipse-säätiö perusti yrityksen, jonka tarkoituksena on edustaa yhteisöä. Yritys luotiin puolueettomuuden ja avoimuuden turvaamiseksi. (Eclipse)

6.2 RoboExplorer

RoboExplorer on robotin ohjausohjelmisto, joka on tarkoitettu Mitsubishin valmistamille, CR-sarjan ohjausyksikköä käyttäville roboteille. Se sisältää samat toiminnot kuin ohjausyksikkö ja opetusyksikkö eli sillä voidaan mm. kirjoittaa MELFA BASIC -ohjelmia, asettaa koordinaatteja ohjelmille, ohjata robottia ja hallinnoida robotin muistiin tallennettuja ohjelmia. (Roboexplorer)

7 TOTEUTUS

7.1 Taustatiedot

Ladontarobotti on kytketty Ethernet-väylän kautta robotin häkin vieressä olevaan tietokoneeseen. Tietokone puolestaan on kytketty koulun sisäiseen verkkoon.

7.2 Suunnittelu

Työn suunnittelu oli suhteellisen lyhyt prosessi. Koska ei löytynyt toista samankaltaista ohjelmaa, jota olisi voinut käyttää suunnittelussa hyödyksi, sovellettiin yksinkertaista serversocket-sovellusta tiedon siirtämiseen ohjausmoduulilta robotille. Ohjelma koostuu kolmesta pääosasta: Ohjausmoduuli eli käyttöliittymä, palvelin ja kontrolleri, joka lähettää palvelimelta saadut käskyt robotille. Jotta ylimääräisiltä luokilta vältytäisiin, palvelimen asiakaspääte sulautettiin käyttöliittymän servlet-osioon.

Käyttöliittymä suunniteltiin muistuttamaan robotin fyysisiä ohjausyksiköitä. Se toteutettiin käyttämällä Html- ja Java Servlet -ohjelmointia.

Ohjelmointikieleksi valittiin Java, koska se soveltui parhaiten verkkosivun kautta toimivaan ohjelmistoon. Lisäksi päätökseen vaikutti oma kokemus javan käytöstä.

7.3 Toiminta

Robotin ohjaimeen pääsee samalta sivustolta kuin minne robotin webkameroiden lähettämä kuva välitetään. Videoohjelmiston alapuolella sijaitsee linkki, joka avaa popup-tyyppisen ikkunan, joka sisältää ohjaimen. Ohjain tulee ensin yhdistää robotille painamalla servojen käynnistys painiketta. Tämä luo yhteyden ohjaimen ja robotin välille. Kun client on yhdistetty, muut eivät voi luoda yhteyttä robotille. Tällä tavalla vältetään ongelmat, joita monen käyttäjän yhtäaikainen toiminta voisi tuottaa.

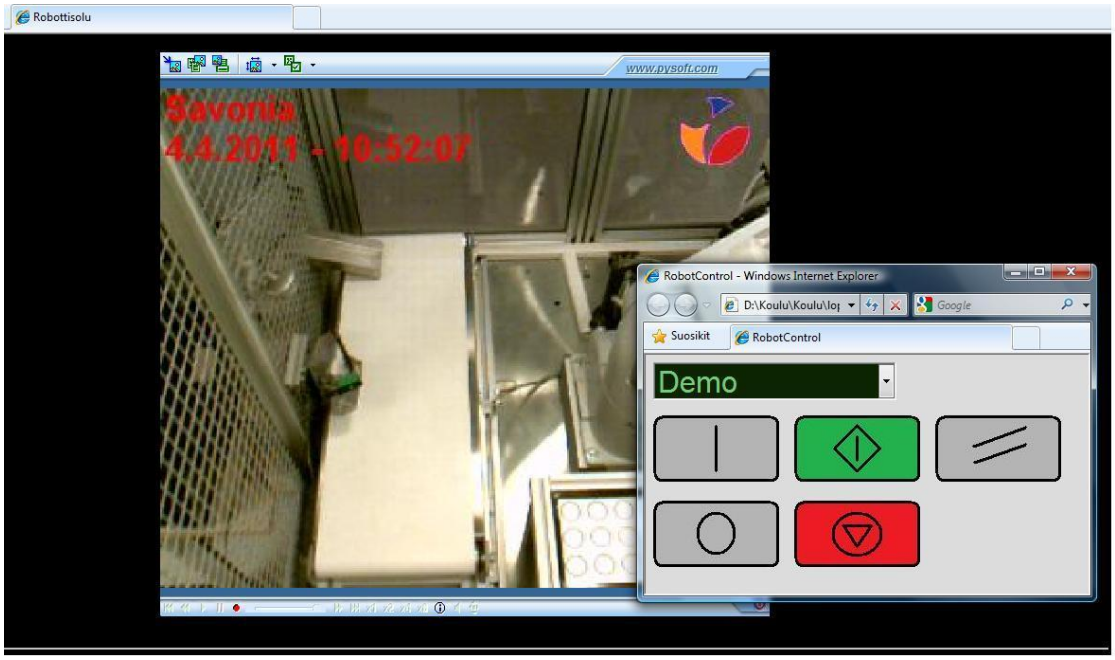
Ohjain sisältää painikkeet ohjelman ajamiselle, pysäyttämiselle sekä robotin nollaamiselle. Lisäksi ohjain sisältää ohjelman valitsemista varten luetteloruudun. Luettelosta valitaan ohjelma ja painetaan suoritus painiketta. Myös omien ohjelmien suorittaminen onnistuu, jos ne ovat tallennettu robotin muistiin. Webkameroiden lähettämän kuvan kautta nähdään robotti, kun se suorittaa ohjelmaa. Tarvittaessa voidaan painaa pysäytys painiketta jolloin ohjelman suoritus pysähtyy. Jos tämän jälkeen painetaan suoritus painiketta, robotti jatkaa

ohjelman suoritusta. Jos halutaan suorittaa joku toinen ohjelma, tulee ensin painaa resetointi painiketta ja sitten valita uusi ohjelma.

Yhteys suljetaan sammuttamalla servot. Lisäksi yhteys katkaistaan automaattisesti, jos asiakas sulkee ohjausikkunan.

7.4 Käyttöliittymä

Ohjaimen käyttöliittymä koostuu html-sivusta ja Java Servlet -luokasta. Html-sivu koostuu yhdestä html-formista, joka sisältää useita button-tyyppisesta elementtejä, yhden select-tyyppisen elementin ja yhden tekstikentän. Jokainen button-elementti on submit-tyyppinen ja jokainen niistä kutsuu servlettiä. Sivu sisältää pienen Javascript-osion, joka näyttää tai piiloittaa tekstikentän riippuen luettelon arvosta. Jos luettelon valittu arvo on Oma, tekstikenttä tulee näkyviin. Muilla arvoilla tekstikenttä on piilossa.



Kuva 5: Ohjelmiston käyttöliittymä

7.4.1 Servlet

Servlet sisältää käyttöliittymän dynaamisen sisällön. Asiakkaan puoleinen socket on sulautettu doPost-metodin sisään. Jokaiselle elementille luodaan muuttuja, joihin haetaan arvot käyttämällä request.GetParameter()-metodia. Kun html kutsuu servlettiä, servlet valikoi toiminnon riippuen siitä, mitä painiketta html-sivulla on painettu. Painettu button saadaan selvitettyä tarkastelemalla painikkeiden palauttamia arvoja. Se painike, jonka palauttama arvo ei ole null, on painettu painike. Toiminnon valinta suoritetaan

yksinkertaisilla ehtolauseilla. Toiminto tallennetaan muuttujaan, joka välitetään serverille käyttäen OutputWriteria.

Mahdolliset virheilmoitukset robotilta tulevat InputReaderilla, ja ne keskeyttävät ohjelman toiminnan virheilmoituksella. Jos robotti lähettää virheen, sen jälkeen annettuja käskyjä ei suoriteta ennen kuin robotti on nollattu.

7.5 Palvelin

Palvelimena käytetään yksinkertaista server-client socket- palvelinta.

7.5.1 Asiakaspääte eli ClientSocket

Asiakaspääte eli clientsocket koostuu viidestä osasta: Käyttöliittymä, yhteyden muodostus, viestien lähettäminen serverille, viestien vastaanotto ja yhteyden sulkeminen. Asiakaspääte on koodattu servlettiin, jotta ylimääräisiltä luokilta vältyttäisiin.

Yhteys muodostetaan avaamalla portti ohjelman käyttöön. Yhteyttä varten haetaan serverin sijainti käyttämällä inetaddress-muuttujaa, jonka parametrinä on joko serverin ip-osoite, www-osoite tai nimi (esim.localhost). Tässä tapauksessa käytetään serverin ip-osoitetta. Ip-osoitetta käytettäessä voidaan käyttää kahta eri funktiota: GetByName(), jonka parametriksi tulee ip-osoite kirjallisessa muodossa ,tai GetByAddress(), jonka parametriksi tulee ip-osoite bittimuodossa. Molemmilla tavoilla on käyttötarkoituksensa, mutta tässä tapauksessa käytetään GetByName()-metodia.

InetAddress Addr = InetAddress.getByName("10.2.35.43");

Yhteys muodostetaan käyttämällä Socket-muuttujaa, jonka parametrinä ovat edellä mainittu InetAddress-muuttuja ja portin numero. Portti, johon otetaan yhteyttä, on 7070, jonka palvelin osa on varannut ohjelman käyttöön.

Socket conn =new Socket(Addr,7070);

Viestien lähettämistä varten luodaan BufferedOutputStream,jonka kautta data kulkee serverille. BufferedOutputStream on suojattujen viestien lähetykseen käytettävä stream eli ”virta”. Lisäksi asetetaan OutputStreamWriter, joka muuntaa bittimuotoisen datan merkkimuotoon. Viestit lähetetään komennolla out.write().

```
BufferedOutputStream strmout = new
BufferedOutputStream(conn.getOutputStream());
OutputStreamWriter out = new OutputStreamWriter(strmout, "US-ASCII");
out.write(output);
```

Viestien vastaanottamista varten tarvitaan saman kaltaiset funktiot kuin lähettämistä varten. BufferedInputStream luo viestintäyhteyden serveriin ja InputStreamReader kääntää datan luettavaan muotoon. Viestit otetaan vastaan in.read() –metodilla ja viestit tallennetaan StringBuffer-tyyppiseen muuttujaan merkki kerrallaan.

```
BufferedInputStream strmin = new
BufferedInputStream(conn.getInputStream());
InputStreamReader in = new InputStreamReader(strmin, "US-ASCII");
int c;
while ( (c = in.read()) != -1){
                instr.append( (char) c);
}
```

Yhteys suljetaan komennolla connection.close().

7.5.2 Palvelinpääte eli ServerSocket

Palvelinpääte eli server-socket koostuu neljästä osasta: Serverin pystyttäminen, yhdistäminen clienttiin, kommunikointi ja yhteyden sulkeminen.
Serveri pytsytetään luomalla kaksi muuttujaa, ServerSocket ja Socket.

```
static ServerSocket srvrsocket;
static Socket conn;
```

ServerSocketilla avataan portti ohjelman käyttöön. Portin valinta riippuu käyttökohteesta. Portit 1-1023 on varattu mm. http ja ftp-yhteyksille. Local host eli paikalliselle verkolle käytetään yleisimmin porttia 8080. Portiksi valittiin portti 7070. Tietoliikenne sallitaan portin läpi reitittimen asetuksia säätämällä.

```
srvrsocket = new ServerSocket(7070);
```

Socket muuttujalla luodaan yhteys clienttiin käyttämällä avatun serversocketin funktiota accept(). Accept()-metodi käytännössä pysäyttää ohjelman kulun ja jää odottamaan clientin yhdistämistä.

conn = srvrsocket.accept();

Kommunikointi toimii samaan tapaan kuin asiakaspäätteessä. Serverille luodaan BufferedOutputStream ja OutputStreamWriter datan lähettämistä varten ja BufferedInputStream ja InputStreamReader datan vastaanottamista varten. Clientilta saadut viestit välitetään kontrollerille, joka muuntaa datan käskyiksi ja välittää käskyt robotille. Kommunikointi kontrollerin kanssa suoritetaan kontrollerilla sijaitsevan metodin kutsulla.

Serveri sulkee yhteyden sockettiin, kun client sulkee yhteyden.

7.6 Kontrolleri

Kontrolleri välittää saadut tiedot käskyinä robotille. Kontrolleri yhdistetään robottiin samaan tapaan kuin client yhdistetään serveriin. InetAddress.GetByName()-komennolla valitaan robotin ip-osoite ja Socket-komennola avataan yhteys oikean portin kautta. Robotin käyttämä ip on 192.168.0.1 ja portti on 10001, joten yhteydenotto koodissa näyttää tällaiselta:

InetAddress robotaddr = InetAddress.getByName("192.168.0.1");
*Socket conn = **new** Socket(robotaddr, 10001);*

Käskyt välitetään robotille käyttämällä OutputStream().write() –komentoa. Mahdolliset virhe-ilmoitukset ja muu robotin lähettämä data vastaanotetaan InputStream –funktion avulla. Puskuroinnille ei tässä välissä ole tarvetta lyhyen välimatkan vuoksi. Data ei ehdi vahingoittua, ellei johdoissa ole mitään vikaa.

Robotille välitetyt viestit ovat lyhyitä ja yksinkertaisia. Esimerkiksi ohjelma suoritetaan lähettämällä robotille komennolla START_p ja pysäytetään komennolla STP. Ajettava ohjelma valitaan lähettämällä komento PN=$$, missä $$ on ohjelman nimi. Komennot muutetaan robotin käyttämään ASCII –muotoon ennen kuin ne lähetetään robotille.

8 PÄÄTÄNTÄ

Robottien yleistyessä myös niiden tarvitsemien ohjelmistojen tarve kasvaa. Etäohjauksen avulla voitaisiin hallita robotteja laajemmalla alalla, jopa maailmanlaajuisesti. Etsiessäni minkäänlaisia etäohjausohjelmistoja, en löytänyt kuin muutamia.

Työ oli selvästi haastavin projekti koko opintoaikana. Oma kokemus ohjelmiston kehityksestä oli aikaisemmin erittäin vähäistä, joten totuttelua oli paljon. Erinäiset kurssit antoivat suhteellisen hyvät lähtökohdat projektille, mutta java ohjelmointikielenä on hyvin laaja ja joissain tilanteissa monimutkainen. Koska tällaiset ohjelmistot ovat harvinaisia, oli oikean rakenteen löytäminen vaikeaa. Lopulta googlen avulla sopiva ratkaisu löytyi ja työ pääsi jatkumaan.

Ohjelmisto on vielä suhteellisen yksinkertainen ja siinä on joitain parantelu mahdollisuuksia. Esimerkiksi varsinainen robotin asennon ohjaus, jonka valitettavasti jouduin hylkäämään liian suurten riskien vuoksi, olisi hyvä ominaisuus. Törmäysten estämiseksi tarvitsisi kartoittaa robotin koko toimintasäde ja asettaa robotin liikkumiselle rajat työkaluilla ja ilman. Lisäksi pitäisi löytää ratkaisu ongelmaan, joka syntyy viiveiden kasaantumisesta. Ohjelmistoni toimisi hyvänä pohjana paljon paremmalle ja monipuolisemmalle etäohjausohjelmistolle.

LÄHTEET JA TUOTETUT AINEISTOT

Ouman Oy, Eh-net[viitattu 18.4.2013] http://www.ouman.fi/fi/eh-net/

Isom, James. A brief history of Robotics, [viitattu 18.4.2013], http://robotics.megagiant.com/history.html

Craig, John J. 2005. Introduction to Robotics Mechanics and Control.

Mitsubishi Electric Company. 2000. Standard Specifications Manual.

Kinnunen, Jukka. Syksy 2008. JAVA-OHJELMOINTI [Luento]. Varkaus: Savonia Ammattikorkeakoulu.

Kinnunen, Jukka. Servlets www-ohjelmointia Javalla. Kurssiaineisto.

Eclipse [viitattu 18.4.2013] saatavissa: http://www.eclipse.org

RoboExplorer [viitattu 18.4.2013] saatavissa: http://roboexplorer.iwdownload.com/

LIITE 1

Ladontarobotin etäohjausohjelman koodi:

Html-käyttöliittymä:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-31j">
<title>RobotControl</title>

<SCRIPT LANGUAGE="JavaScript">
function showTxtBox() {
if(document.form1.ProgramLstBox.value=="ownd"){
document.form1.Program.type = 'text';
}
else{
document.form1.Program.type = 'hidden';
}
}
</SCRIPT>

</head>
<body style="background-color: #DCDCDC">
<form name="form1" method="post" action="RoboServlet">
<select name="ProgramLstBox" style="width: 257px; height: 40px;font-size: xx-large; back-
ground-color: #0C2400; color: #75CC77;" onchange="showTxtBox()">
<option selected="selected" value="Demo" >Demo</option>
<option value="HG" >Hand Grab</option>
<option value="HD" >Hand Drop</option>
<option value="VG" >Vacuum Grab</option>
<option value="VD" >Vacuum Drop</option>
<option  value="ownd" >Oma</option>
</select>
<p><input type="hidden" name="Program" value="" style="width: 230px; height: 40px; font-
size: xx-large; background-color: #0C2400; color: #75CC77;">

<p><input type="image" src="imgs\svoonbtn.jpg" width="133" height="70"
alt="Submit"name="son" value="svoon" />
<input type="image" src="imgs\startbtn.jpg" width="133" height="70" alt="Submit"
name="start" value="run" />
<input type="image" src="imgs\resetbtn.jpg" width="133" height="70" alt="Submit" name="rst"
value="reset" />
<p><input type="image" src="imgs\svooffbtn.jpg" width="133" height="70" alt="Submit"
name="soff" value="svooff" />
<input type="image" src="imgs\stopbtn.jpg" width="133" height="70" alt="Submit" name="stop"
value="end" />
</form>
</body>
</html>
```

Servlet/ClientSocket:

```
import java.net.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
import javax.swing.*;

@WebServlet("/RoboServlet")
```

```
public class RoboServlet extends HttpServlet {
private static final long serialVersionUID = 1L;

public RoboServlet() {
super();
// TODO Auto-generated constructor stub
}

public void ShowDialogBox(String err){          //Ilmoitusikkuna virheille
JFrame frame;
frame = new JFrame("Error: "+err);
JButton button = new JButton("Ok");
frame.add(button);
frame.setSize(400, 400);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private Socket Connector(){                     //Yhdistys serversocketille
try{
InetAddress Addr = InetAddress.getByName("10.2.35.43");
Socket conn =new Socket(Addr,7070);
return conn;
}catch(IOException e){
ShowDialogBox(e.toString());
return null;
}
}

private String Communicator(String output, Socket conn) {   //Kommunikointi
String inp=null;
StringBuffer instr = new StringBuffer();
if(conn!=null){
try {
BufferedOutputStream strmout = new BufferedOutputStream(conn.getOutputStream());
OutputStreamWriter out = new OutputStreamWriter(strmout, "US-ASCII");

out.write(output);
out.flush();

BufferedInputStream strmin = new BufferedInputStream(conn.getInputStream());
InputStreamReader in = new InputStreamReader(strmin, "US-ASCII");

int c;
while ( (c = in.read()) != -1){
instr.append( (char) c);
}

inp=instr.toString();

} catch (IOException e) {
ShowDialogBox(e.toString());
}
}
else{
ShowDialogBox("No Socket Connection");
}
return inp;
}

private void DisConnector(Socket conn){         //Yhteyden katkaisu
try {
conn.close();
} catch (IOException e) {
```

```
ShowDialogBox(e.toString());
}
}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Ser-
vletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws Ser-
vletException, IOException {
//Luodaan session, johon yhteys tallennetaan
HttpSession session = request.getSession();

String resp="";
Socket connection = (Socket) session.getAttribute("Socket Connection");
if(connection==null){
session.setAttribute("Socket Connection", connection);
}
String prgrm=request.getParameter("ProgramLstBox");
String prgrmown=request.getParameter("Program");
String servoon=request.getParameter("son");
String start=request.getParameter("start");
String reset=request.getParameter("rst");
String servooff=request.getParameter("soff");
String stop=request.getParameter("stop");
//Jos valittu ohjelma on oma
if(prgrm=="ownd"){
prgrm=prgrmown;
}
//Valitun painikkeen etsiminen
if(servoon!=null){
if(connection==null){   //Jos yhteys on jo luotu, ei muodosteta uutta
connection=Connector();
}
servoon=null;
}
else if(start!=null){
resp=Communicator("RUN "+prgrm +"    ",connection);
start=null; //Lähetetään myös valittu ohjelma
}
else if(reset!=null){
resp=Communicator("RESET",connection);
reset=null;
}
else if(servooff!=null){
if(connection!=null){   //Ei katkaista olematonta yhteyttä
DisConnector(connection);
}
servooff=null;
}
else if(stop!=null){
resp=Communicator("STOP",connection);
stop=null;
}
else{}

if(resp!=""){           //Näytetään mahdolliset robotin lähettämät viestit
ShowDialogBox(resp);
}           //Palataan takaisin käyttöliittymään
response.sendRedirect("http://wilma.savonia-
amk.fi/roboprojektit/robottisolu/webcam/RoboClient.html");
}
}
```

ServerSocket:

```
import java.net.*;
import java.io.*;

public class SocketServer {

static ServerSocket srvrsocket;
static Socket conn;

public static void main(String[] args) {
StringBuffer instr = new StringBuffer();
String inp;
String output="";
CommunicatorSocket Comm=new CommunicatorSocket();
int c;
try {
srvrsocket = new ServerSocket(7070);
while(true){
conn = srvrsocket.accept();

BufferedInputStream strmin = new BufferedInputStream(conn.getInputStream());
InputStreamReader in = new InputStreamReader(strmin, "US-ASCII");

while ( (c = in.read()) != -1){
instr.append( (char) c);
}

inp=instr.toString();
output=Comm.Communicator(inp);      //Kutsu kontrollerilla olevaan funktioon,
                                    //joka palauttaa robotin viestit
BufferedOutputStream strmout = new Buffere OutputStream(conn.getOutputStream());
OutputStreamWriter out = new OutputStreamWriter(strmout, "US-ASCII");

out.write(output);
out.flush();
}
} catch (IOException e) {
System.out.println("IOException: "+e);
}

}

}
```

Kontrolleri:

```
import java.net.*;
import java.io.*;

public class CommunicatorSocket {
            //Kääntää viestit robotin ymmärtämiksi käskyiksi
public String Translator(String word){
String ordr="";
if(word.substring(0,2)=="RU"){
ordr="START_P"+word.substring(6,4);;
}
else if(word.substring(0,2)=="ST"){
ordr="STP";
}
else if(word.substring(0,2)=="RE"){
ordr="RS";
}
```

```
return ordr;
}
public String Communicator(String comm){
String output="", inp=null;
StringBuffer instr=new StringBuffer();
int c;
try {
InetAddress robotaddr = InetAddress.getByName("192.168.0.1");
Socket conn = new Socket(robotaddr, 10001);

OutputStream strmout = new OutputStream( conn.getOutputStream());
OutputStreamWriter out = new OutputStreamWriter(strmout, "US-ASCII");

output=Translator(comm);
//Jos käsky on ohjelman suoritus, lähetetään ensin ohjelma
if(output.substring(0, 3)=="STA"){
out.write(output.substring(7,4));
out.flush();
try {
Thread.sleep(2000);     //odotetaan että robotti ehtii käsitellä viestin
}catch(Exception e){
}
output=output.substring(0,6);
}

out.write(output);
out.flush();

InputStream strmin = new InputStream(conn.getInputStream());
InputStreamReader in = new InputStreamReader(strmin, "US-ASCII");

while ( (c = in.read()) != -1){
instr.append( (char) c);
}

inp=instr.toString();
conn.close();
} catch (IOException e) {
System.out.println("IOException: "+e);
}
return inp;
}

}
```